# Параметры и возвращаемое значение функции main

Все программы, написанные на C, содержат функцию main. При запуске программы это будет первая пользовательская функция, исполняемая в контексте процесса (но не первая функция вообще, так как до main будут выполнены процедуры инициализации, зависящие от операционной системы и используемых библиотек C).

Функция main получает параметры от операционной системы (точнее, от оболочки (shell), из которой запускается программа). Возвращаемое значение функции main считывается оболочкой и в командной строке доступно через переменную $?.

**Внимание!** Через $? доступен только младший байт возвращаемого значения main

Есть несколько определений функции main в библиотеке C:

int main() – сокращенное определение без аргументов

int main(int argc, char \*argv[]) – стандартное определение

int main(int argc, char \*argv[], char \*envp[]) – расширенное определение, не входит в стандарт ANSI C, доступно в UNIX, UNIX-подобных ОС и Windows.

Здесь

argc – число аргументов, переданное в командной строке при запуске программы

char \*argv[] – массив указателей на строки со значениями аргументов командной строки, содержит argc элементов

char \*envp[] – массив указателей на строки с именами и значениями переменных окружения в виде ИМЯ=ЗНВЧЕНИЕ (подробнее о стандартных переменных окружения UNIX можно посмотреть здесь: <https://www.gnu.org/software/libc/manual/html_node/Standard-Environment.html>). Число переменных окружения не передается в main. Массив envp[] нужно читать последовательно с начала, пока не будет прочитано значение NULL.

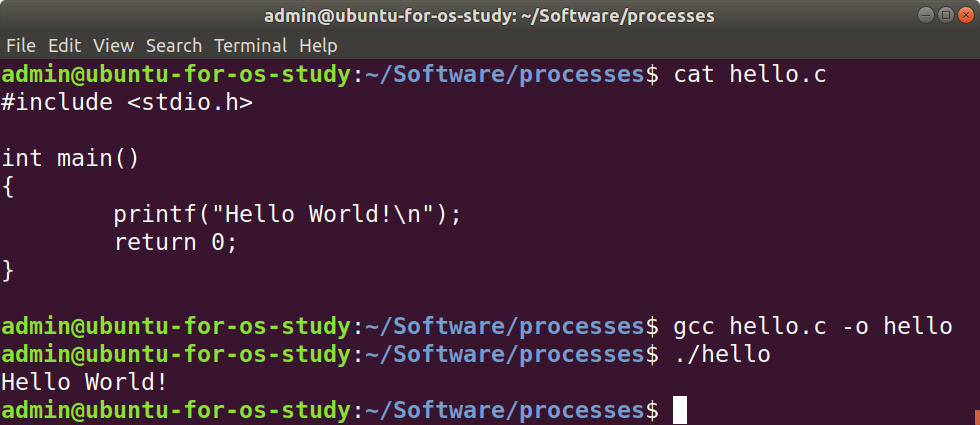
Переменные окружения можно устанавливать из командной строки командой export:

export MY\_ENV\_VAR=”Моя переменная окружения”

# Компиляция программы из командной строки

В UNIX и UNIX-подобных системах обычно используются компиляторы GNU. Запускается командой gcc (сокращение от Gnu Compilers Collection). Нужный компилятор выбирается по расширению файла, поэтому для программы на C расширение .c является обязательным. Опция -o позволяет задать имя выходного файла.

Пример:



# Разбор аргументов командной строки

Существует общепринятый синтаксис командной строки, позволяющий задавать опции работы программы. Аргументы командной строки разделяются пробелами (если значение параметра включает пробел, то такое значение нужно взять в кавычки). Синтаксис опции следующий:

-<код опции>[значение опции]

Опцией является аргумент командной строки, начинающийся с символа ‘-‘. Опция может находиться в произвольном аргументе командной строки (не обязательно опции должны начинаться с первого аргумента, не обязательно опции должны идти в командной строке подряд (между ними могут быть аргументы, которые опциями не являются), после опций могут быть указаны еще аргументы).

Специальная опция ‘--’ принудительно завершает список опций, даже если в командной строке есть еще аргументы.

Если опция предполагает установку значения, то его нужно указать после ключа (значение можно отделить от ключа пробелом, но это не обязательно).

Опции без параметров можно указывать без пробела за одним ‘-‘, например, ls -l -a -i и ls -lai приведут к одинаковому результату.

Порядок следования опций не имеет значения, например, ls -lai ls -ail ls -lia приведут к одинаковому результату.

В библиотеке C существует специальная функция getopt для работы с опциями (определена в unistd.h):

int getopt(int argc, char \* const argv[], const char \*optstring)

В функцию нужно передать аргументы main и строку опций, определяющую список опций. Символ ‘:’ после опции означает наличие у опции значения, два двоеточия ‘::’ означает, что значение является опциональным и может не указываться. Например “ab:c::” - определены три опции: -a, -b и -c. Опция -a используется без значения, опция -b требует обязательно указать значение, опция -c может использоваться как со значением, так и без значения.

Если значение опции должно передаваться обязательно, то любая строка после кода опции будет трактоваться как значение (даже если она начинается с ‘-‘). Если значение опции может передаваться опционально, то любая строка после кода опции, введенная без пробела после кода опции, будет трактоваться как значение (даже если она начинается с ‘-‘). Если же ввести пробел после кода опции, то следующая строка будет трактоваться как параметр, только если она не распознается как опция.

Функция getopt использует глобальные переменные для управления и возвращения значений параметров опций:

optarg - указатель на строку со значением параметра опции или NULL, если нет параметра

optind - индекс текущей опции в массиве argv. Если в строке параметров встретится специальная опция ‘--’, то в переменной optind будет индекс аргумента, следующего за ‘--’

Документацию по getopt см. <https://man7.org/linux/man-pages/man3/getopt.3.html> или man 3 getopt в командной строке.

Пример:

#include <unistd.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

while (1)

{

int c = getopt(argc, argv, "ab:c::d");

if (c == -1)

{

break;

}

switch (c)

{

case 'a':

printf("option 'a'\n");

break;

case 'b':

printf("option 'b' with '%s'\n", optarg);

break;

case 'c':

if (optarg)

{

printf("option 'c' with '%s'\n", optarg);

}

else

{

printf("option 'c' witout argument\n");

}

break;

case 'd':

printf("option 'd'\n");

break;

}

}

return 0;

}

## Длинные опции

Традиционные опции обозначаются одним символом (код опции), что не всегда удобно при работе (трудно запомнить называния однобуквенных кодов), удобнее было бы обозначать коды опций осмысленными словами. Такую возможность дает функция getopt\_long (определена в getopt.h)

int getopt\_long(int argc, char \* const argv[], const char \*optstring, const struct option \*longopts,  
 int \*longindex);

Длинные опции начинаются с двух дефисов ‘--‘.

В функцию getopt\_long нужно передать аргументы main. Строка опций optstring аналогична рассмотренной в getopt. Массив longopts описывает длинные опции.

Элементами массива longopts являются экземпляры структуры option, определенной следующим образом:

struct option {  
 const char \*name; // название опции  
 int has\_arg; // наличие аргумента: no\_argument, required\_argument, optional\_argument  
 int \*flag; // =NULL  
 int val; // возвращаемое значение getopt\_long, если будет найдена данная опция  
};

Название опции может содержать пробелы, но тогда при использовании опции ее нужно писать в кавычках.

Последний элемент массива longopts должен быть заполнен нулями (все поля структуры option нужно обнулить).

Параметр longindex может быть NULL

Пример использования getopt\_long:

#include <getopt.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

struct option longopts[] =

{

{

.name = "advanced",

.has\_arg = no\_argument,

.flag = NULL,

.val = 'a'

},

{

.name = "base address",

.has\_arg = required\_argument,

.flag = NULL,

.val = 'b'

},

{

.name = "cursor",

.has\_arg = required\_argument,

.flag = NULL,

.val = 'c'

},

{

.name = "debug",

.has\_arg = no\_argument,

.flag = NULL,

.val = 'd'

},

{

}

};

while (1)

{

int c = getopt\_long(argc, argv, "ab:c::d", longopts, NULL);

if (c == -1)

{

break;

}

switch (c)

{

case 'a':

printf("option 'a'\n");

break;

case 'b':

printf("option 'b' with '%s'\n", optarg);

break;

case 'c':

if (optarg)

{

printf("option 'c' with '%s'\n", optarg);

}

else

{

printf("option 'c' witout argument\n");

}

break;

case 'd':

printf("option 'd'\n");

break;

}

}

return 0;

}